Microsoft

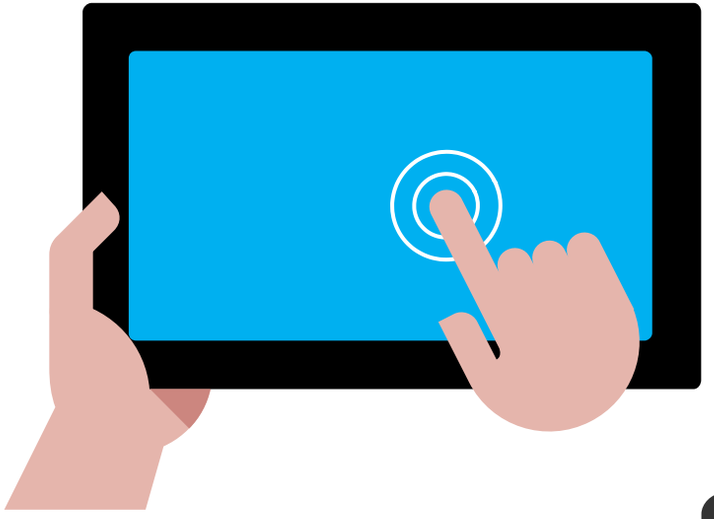
JavaScript Coding for the Touch Interface, Device and Operating System Resources, and More

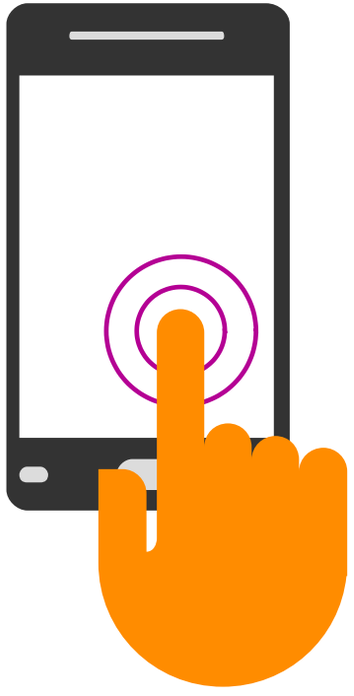
4.5. Respond to the touch interface
4.6. Code additional HTML5 APIs
4.7. Access device and operating system resources

Agenda

1	Touch Interfaces
2	HTML5 APIs
3	Web Storage API

Touch Interfaces



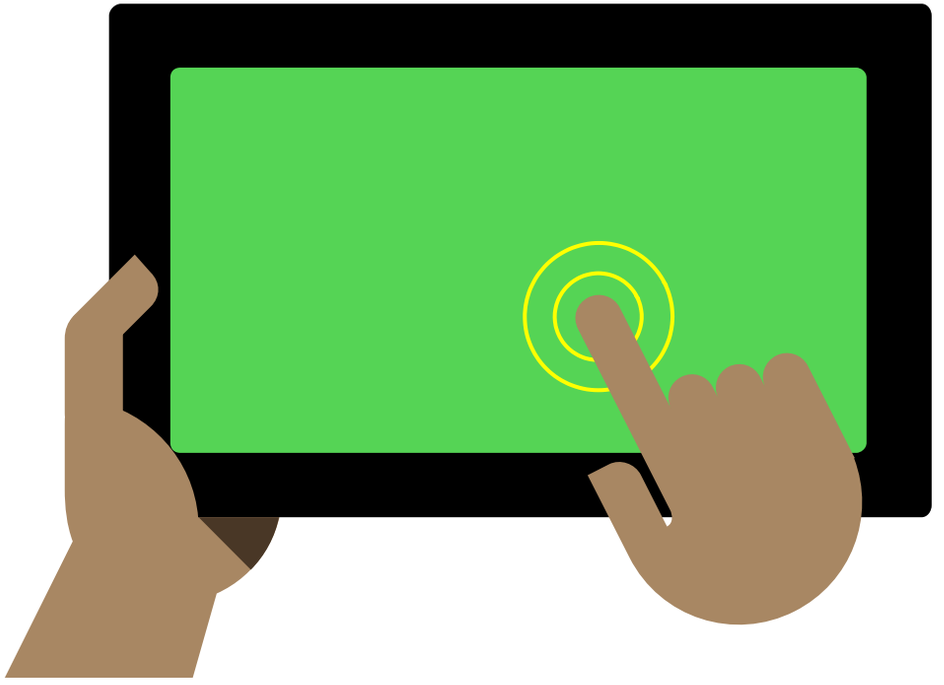


Touch Interfaces

- Devices with **touch interfaces** have screens that are developed specifically for sensing touch
- There are two different types of touch screens:
  - **Capacitive touch screens**
  - **Resistive touch screens**
- Processors treat touches like mouse gestures and relay that information to the operating system and application

Developing for Touch Interfaces

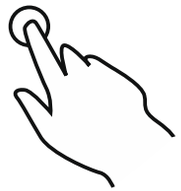
- Before we develop touch-enabled apps, we have to understand how fingers interact with touch-screen devices
- Finger moves are called **gestures**, and can include a press, tap, slide, and more
- How an application responds to a gesture is called a **touch event**
- We can use JavaScript, including the `touchstart`, `touchend`, and `touchmove` methods to create touch events



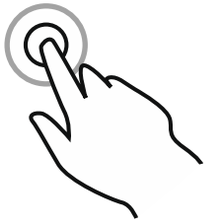
Common Touch Gestures

GESTURE	MOUSE EQUIVALENT	DESCRIPTION
Tap	Left-click	Tap a finger on the screen
Double tap	Left double-click	Tapping a finger on the screen twice
Press and hold	Right-click	Press and hold a finger on the screen, then release
Selection/drag	Mouse drag (selection)	Drag a finger to the left or right
Zoom	CTRL + mouse wheel forward or backwards	Pinch an object inwards or outwards

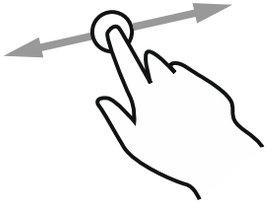
Common Touch Gestures



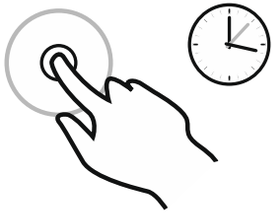
TAP



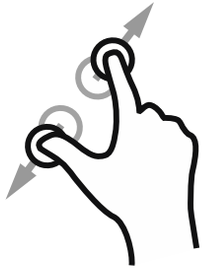
DOUBLE TAP



DRAG



PRESS AND HOLD



ZOOM

Illustrations provided by GestureWorks® (www.gestureworks.com)

Event Listeners

- To make an application respond, we use the `addEventListener` method to attach an event handler to an HTML element
- The general syntax for `addEventListener` is as follows:

```
document.addEventListener("touchstar
t",
  touchStartHandler, false);
```

Gesture Events

- A gesture event occurs when a user touches the screen with multiple fingers
- Common gesture events are:

GESTURE EVENT	DESCRIPTION
`gesturestart`	Every new two-finger gesture triggers a `gesturestart` event
`gesturechange`	When both of those fingers move around the screen, it triggers a `gesturechange` event
`gestureend`	Lifting both fingers from the screen triggers a `gesturechange` event

Touch Events

- A touch event is an action that an application takes in response to a gesture
- Common touch events are:

TOUCH EVENT	DESCRIPTION
`touchstart`	A new finger touch triggers a `touchstart` event
`touchmove`	A `touchmove` event tracks fingers movements as they occur on screen
`touchend`	Lifting a finger from the screen triggers a `touchend` event
`touchcancel`	A `touchcancel` event occurs when a new application is launched

Touch Objects

- A touch object detects input from touch-enabled devices
- Touch objects are referenced in the **touchlist**
  - touchlists track all points of contact on a touch screen
  - a single finger touch has one entry in a touchlist, while a two finger swipe has two entries
- Each touch object features the properties listed in the table to the right

PROPERTY	DESCRIPTION
`identifier`	an id for the touch object
`target`	the HTML element affected
`clientx`	horizontal positioning relative to the browser
`clienty`	vertical positioning relative to the browser
`pagex`	horizontal positioning relative to the HTML document
`pagey`	vertical positioning relative to the HTML document
`screenx`	horizontal positioning relative to the screen
`screeny`	vertical positioning relative to the screen

Different Touchlists

Each touch event features three different touchlists, as featured in the table below

PROPERTY	DESCRIPTION
`touches`	A list of all touch points currently in contact with the screen
`targetTouches`	A list of touch points currently in contact with the screen and whose `touchstart` event occurred within the same node (inside the same target element as the current target element
`changedTouches`	A list of touch points that caused the current event to be fired; for example, in a `touchend` event, this is the finger that was removed

Detecting a Touch Screen Demo

```
document.addEventListener("DOMContentLoaded", init, false);

function init() {
    var canvas =
        document.getElementById("canvas");
    if ("ontouchstart" in document.documentElement) {
        canvas.addEventListener("touchstart", detect, false);
    }
    else {
        canvas.addEventListener("mousedown", detect, false);
    }
}

function detect() {
    if ("ontouchstart" in document.documentElement) {
        alert("Touch screen device detected!");
    }
    else {
        alert("No touch screen device detected.");
    }
}
```

HTML5 APIs

Web Hypertext Application Technology Working Group

WHATWG is a consortium that was formed by Apple, the Mozilla Foundation, and Opera Software to define and document HTML5

- This organization is different from W3C

The WHATWG HTML5 specification includes the following additional APIs in its HTML5 specification:

- Geolocation
- Web Workers
- WebSockets
- File

Geolocation API

The **Geolocation API** allows developers to access the geographical coordinates of a user's device

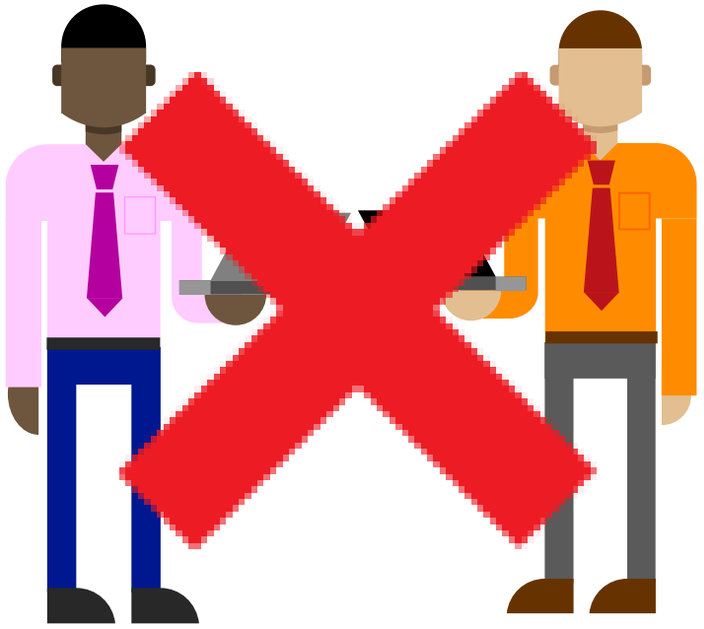
- geographical coordinates include latitude and longitude

The primary functions of geolocation are:

- `getCurrentPosition`: gets a device's current geographic position
- `watchPosition`: tracks changes in position over time, generating events when changes occur

```
navigator.geolocation.getCurrentPosition(showmap);
function showmap(position) {
var latitude = position.coords.latitude;
var longitude = position.coords.longitude;
// Insert code to display a map
}
```

Web Workers



- **Web Workers** are APIs that run scripts in the background
  - They ensure that important actions are performed while users interact with the screen at the same time
- Web Workers pass information through messages
- They run separately from the main HTML document in a separate thread

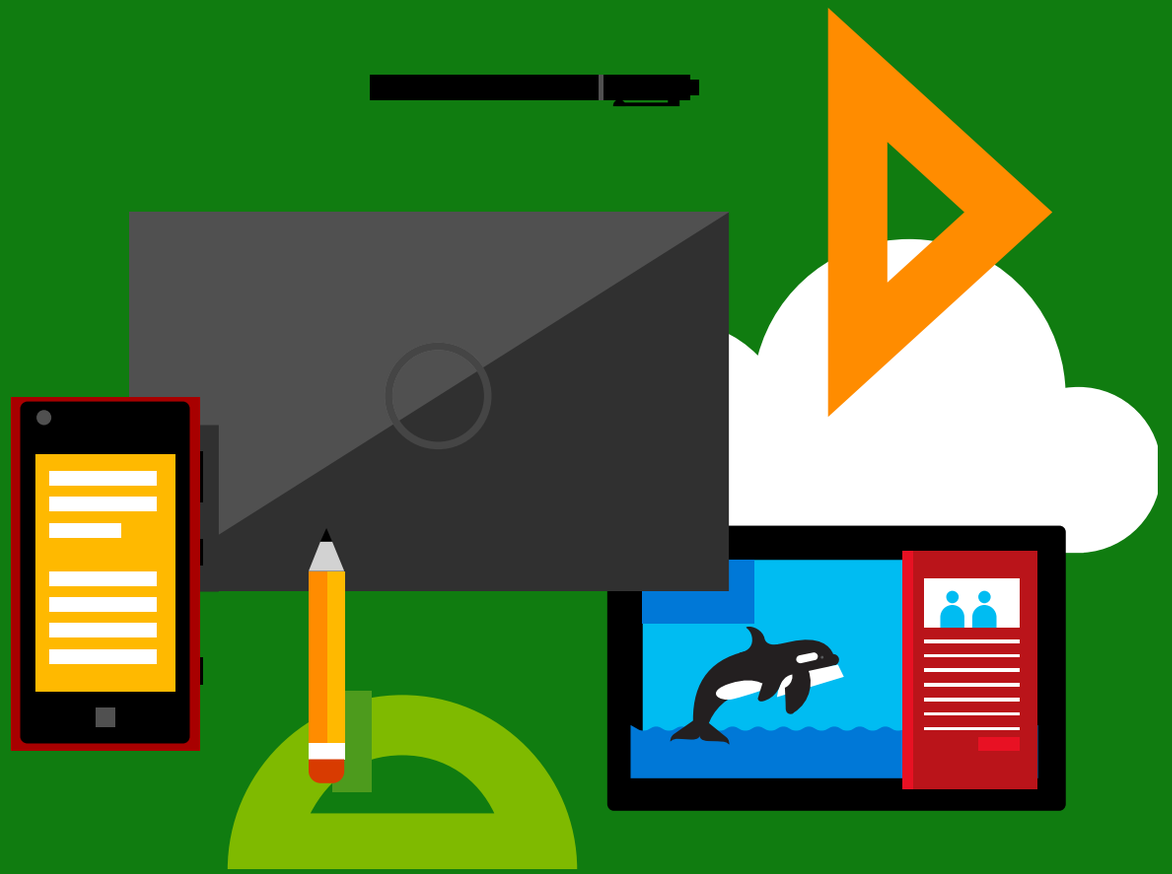
WebSocket API

- The **WebSocket API** creates a simultaneous two-way connection between a client and a Web server
  - this type of connection is called a socket, and it allows data to flow back and forth freely
- WebSockets are commonly used for real-time applications like chat, multiplayer online gaming, or sports score tickers
- There are three primary events that occur with WebSocket communication:
  - `onopen`: when a socket opens
  - `onmessage`: when a message is received from a Web server
  - `onclose`: when a socket closes

File API

- The File API allows Web applications to upload files from local storage to remote servers
- The File API features several interfaces for accessing files, including:
  - **File**: reads in a file as a URL
  - **FileList**: permits upload of multiple files or a folder of files
  - **Blob**: provides access to raw binary data
  - **FileReader**: Provides methods to read and display a file

Web Storage API



Web Storage API

- The Web Storage API allows you to store data in the browser versus a server
- There are two different types of storage:
  - local
  - session

localStorage and sessionStorage

- localStorage lets users save larger amounts of persistent data
  - there is no limit to how long the data persists
- sessionStorage lets users save session state data
  - it only lasts for the duration of the session
- Both objects allow users to store large amounts of data without slowing down a connection because data is transferred only when requested

Implementing Web Storage, pt. 1

Use the `localStorage` and `sessionStorage` objects with the methods in the table to manage key/value pairs

METHODS	FUNCTION
`setItem(key, value)`	Creates a new key/value pair
`getItem(key)`	Gets the value for the specified key
`removeItem()`	Removes the specified key/value pair
`clear();`	Clears all information from the storage object

Implementing Web Storage, pt. 2

Add a key/value pair to a `sessionStorage` object by using the following format:

```
sessionStorage.setItem('key', 'value');
var myVar =
sessionStorage.getItem('key');
```

Add a key/value pair to a `localStorage` object by using the following format:

```
localStorage.setItem('key', 'value');
var myVar = localStorage.getItem('key');
```

Accessing Hardware

- Building with HTML5, CSS, and JavaScript leads to device-independent apps
- Device-independent apps are able to access hardware capabilities, such as:
  - Global Positioning System (GPS)
  - Accelerometer
  - Camera

Summary

1	Touch Interfaces
2	HTML5 APIs
3	Web Storage API